How to Easily Rename Variables in SAS Using the RENAME Statement

Authored by

stats writer

December 1, 2025

The ability to effectively manage and organize data is central to successful data analysis in SAS. One of the most fundamental data management tasks is renaming variables. Renaming is often essential for clarity, conformity to naming conventions, or simply shortening lengthy, cumbersome names. The primary mechanism for achieving this in SAS is through the powerful RENAME statement.

The RENAME statement allows users to assign new names to one or more existing variables within a dataset operation. Its general syntax is deceptively simple: `RENAME oldname1=newname1 oldname2=newname2;`. For instance, transforming an ambiguous variable named `age` into the more descriptive `age_group` is achieved using `RENAME age=age_group;`. This functionality is invaluable for enhancing the readability and interpretability of your final datasets, a crucial step before sharing results or performing complex statistical analysis.

While the basic syntax remains constant, the RENAME statement can be implemented in several contexts within SAS, including inside a DATA Step or within the versatile PROC DATASETS procedure. Understanding where and how to place the statement dictates whether you create a new dataset with the new names, or modify the original dataset in place. We will explore both common methods and advanced bulk renaming techniques in the examples that follow.

You can use the **rename option** directly within the `SET` statement of a DATA Step to rename one or more variables as the dataset is being read and copied. This is the preferred method when creating a new dataset based on an existing one, ensuring that the original source data remains untouched, which is a key best practice in data hygiene.

This function uses the following basic syntax structure within the `SET` statement parentheses:

```
data new_data;
set original_data (rename=(old_name=new_name));
run;
```

This approach is highly efficient because the renaming occurs during the input phase of the new dataset creation. Before diving into the specific renaming techniques, we must establish a working dataset. The following code demonstrates the creation and initial viewing of the example dataset that will be used throughout this tutorial:

```
/*create dataset for demonstration*/
data original_data;
input x y z;
datalines;
1 4 76
```

```
2 3 49
2 3 85
4 5 88
2 2 90
;
run;

/*view the initial dataset structure*/
proc print data=original_data;
```

Obs	x	y	z
1	1	4	76
2	2	3	49
3	2	3	85
4	4	5	88
5	2	2	90

Example 1: Renaming a Single Variable in a DATA Step

The most common renaming task involves changing the name of just one variable. This is typically done when a variable name is non-descriptive or conflicts with a reserved keyword. When using the `SET` statement, the renaming happens only in the context of the dataset being created (`new_data`), leaving the original source dataset (`original_data`) completely intact.

In this example, we aim to rename the variable labeled **x** to **new_x**. This simple modification immediately improves clarity, especially if 'x' represented a critical measurement like 'temperature' or 'score'. We apply the `rename=(x=new_x)` syntax directly inside the parentheses following the source dataset name in the `SET` statement.

The following code shows how to rename just the **x** variable in the dataset:

```
/*rename one variable using the SET option*/
data new_data;
set original_data (rename=(x=new_x));
run;

/*view new dataset to confirm change*/
proc print data=new_data;
```

Obs	new_x	y	z
1	1	4	76
2	2	3	49
3	2	3	85
4	4	5	88
5	2	2	90

Upon viewing the output, notice that **x** has been successfully renamed to **new_x**, while every other variable name (**y** and **z**) remained precisely the same. This method is straightforward and highly recommended for selective, precise renaming operations within the DATA Step flow.

Example 2: Renaming Multiple Variables Simultaneously

When cleaning or preparing large datasets, it is often necessary to rename several variables at once. Fortunately, the syntax for renaming multiple variables is a simple extension of the single-variable method. You continue listing the `old_name=new_name` pairs within the parentheses of the `RENAME` option inside the `SET` statement.

A key syntactical detail to remember is that you **do not** need to include commas, semicolons, or any other separators between the renaming pairs; a space is sufficient. This clean structure contributes to concise and easily readable SAS code, even when dealing with dozens of variable changes.

The following code demonstrates how to rename both the **x** variable to **new_x** and the **y** variable to **new_y** in the dataset:

```
/*rename multiple variables concurrently*/
data new_data_multi;
set original_data (rename=(x=new_x y=new_y));
run;

/*view the resulting dataset*/
proc print data=new_data_multi;
```

Obs	new_x	new_y	z
1	1	4	76
2	2	3	49
3	2	3	85
4	4	5	88
5	2	2	90

The resulting dataset, `new_data_multi`, now reflects the updated names for both variables, illustrating the scalability of the `RENAME` option within the `SET` statement. This method is highly flexible and should be utilized whenever two or more variables require explicit renaming within a single DATA Step.

Advanced Renaming: Using PROC SQL and PROC DATASETS

While the `SET` statement method in the DATA Step is ideal for small, explicit changes, it becomes impractical when you need to rename a large number of variables systematically--for instance, adding a standard prefix or suffix to every variable in the dataset. For these complex, automated tasks, a two-step approach involving PROC SQL and PROC DATASETS is far more efficient.

This advanced technique utilizes PROC SQL to query the SAS metadata dictionary (specifically `dictionary.columns`) and dynamically generate a string of renaming pairs (e.g., `x=_NEWx y=_NEWy z=_NEWz`). This string is stored in a macro variable, which is then passed to the `RENAME` statement within `PROC DATASETS`. The `PROC DATASETS` procedure is crucial here because it allows modification of an existing dataset in place (using the `MODIFY` statement) without the need to recreate the entire dataset, offering significant performance benefits for large files.

Example 3: Add Prefix to All Variables

Suppose you have merged several datasets and need to distinguish the origin of the variables from the `original_data` source. Adding a common prefix, such as `_NEW`, to every variable provides immediate clarity. This process involves concatenating the variable name with the desired prefix using the `CATS` function inside PROC SQL. The result is saved into the macro variable `&list`.

The `WHERE` clause in the SQL query ensures we only target variables within our specific dataset (`WORK.ORIGINAL_DATA`). The subsequent `PROC DATASETS` block then executes the generated RENAME statement using the macro variable. Note that unlike previous examples, this method alters the `original_data` dataset directly.

```
/*define prefix to append to each variable using PROC SQL*/
proc sql noprint;
select cats(name, '=', '_NEW', name)
into :list
separated by ' '
from dictionary.columns
where libname = 'WORK' and memname = 'ORIGINAL_DATA';
quit;

/*add prefix to each variable in dataset using PROC DATASETS*/
proc datasets library = work;
modify original_data;
rename &list;
quit;

/*view updated dataset*/
proc print data=original_data;
```

Obs	NEW_x	NEW_y	NEW_z
1	1	4	76
2	2	3	49
3	2	3	85
4	4	5	88
5	2	2	90

Example 4: Add Suffix to All Variables

Adding a suffix is frequently used when performing transformations or standardizations on existing variables (e.g., transforming `score` into `score_std`). By appending a suffix like **_NEW** to all variables, we quickly denote that these are the modified versions of the data. The logic here is nearly identical to the prefix example, with a slight adjustment to the `CATS` function call to place the suffix after the original variable name.

The following code shows how to add a suffix of **_NEW** to all variables in the dataset:

```
/*define suffix to append to each variable using PROC SQL*/
proc sql noprint;
select cats(name, '=', name, '_NEW')
```

```
into :list
separated by ' '
from dictionary.columns
where libname = 'WORK' and memname = 'ORIGINAL_DATA';
quit;

/*add suffix to each variable in dataset using PROC DATASETS*/
proc datasets library = work;
modify original_data;
rename &list;
quit;

/*view updated dataset*/
proc print data=original_data;
```

Obs	x_NEW	y_NEW	z_NEW
1	1	4	76
2	2	3	49
3	2	3	85
4	4	5	88
5	2	2	90

Summary of Renaming Methods

Choosing the appropriate method for renaming variables in SAS depends entirely on the scope and complexity of the task. For renaming one or a few variables while creating a new dataset, utilizing the `RENAME` option within the `SET` statement in a DATA Step is the most efficient and safest approach, preserving the original data integrity.

For scenarios requiring bulk renaming, such as applying standardized prefixes or suffixes across numerous variables, the combination of PROC SQL (for dynamic name generation) and `PROC DATASETS` (for in-place modification) provides the necessary automation and performance benefits. Mastering both techniques ensures that you can handle any variable naming requirement efficiently within the SAS environment.

Related SAS Data Management Tutorials

The following tutorials explain how to perform other common tasks in SAS: